

Remarks:

Reconsideration of the application, as amended herein, is respectfully requested.

Claims 1 and 4 - 15 are presently pending in the application. Claim 1 has been amended. Claims 2 and 3 were previously canceled. New claims 13 - 15 were added.

In item 4 of the above-identified Office Action, claims 1 - 9 and 11 were rejected under 35 U.S.C. § 103(a) as allegedly being obvious over U. S. Patent No. 6,499,096 to Suzuki ("SUZUKI") in view of U. S. Patent No. 5,913,925 to Kahle et al ("KAHLE") and further in view of U. S. Patent No. 6,324,639 to Heishi et al ("HEISHI"). In item 17 of the above-identified Office Action, claim 12 was rejected under 35 U.S.C. § 103(a) as allegedly being obvious over SUZUKI, KAHLE and HEISHI, and further in view of U. S. Patent No. 6,404,752 to Allen ("ALLEN").

Applicant notes that item 4 of the Office Action indicates that claims 1 - 9 and 11 were rejected under 35 U.S.C. § 103. However, Applicant's claims 2 and 3 were previously canceled from the present case. Further, the only reference to Applicant's claim 10, occurs in item 13 of the Office Action, which starts out by stating, in part:

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

13. Referring to claims 9 and 11, Suzuki has taught wherein the branching control unit comprises:
[emphasis added by Applicant]

As such, Applicant assumes that claims 1 and 4 - 11, and not claims 1 - 9 and 11, have been rejected over **SUZUKI, KAHLE** and **HEISHI**, which rejections (in addition to the rejection of claim 12) Applicant respectfully traverses.

More particularly, Applicant's invention differs from the prior art, among other reasons, in that it uses two different kinds of parallel processing to determine parallelism.

In the prior art, there are known systems that each use only one type of parallel processing determine parallelism. For example, as disclosed in the background of the invention of the instant application, page 2, line 17 - page 3, line 14, known types of processors use one of two methods for determining parallelism. More particularly, in the first type, instruction level parallelism (ILP) processors detect parallelism **during the transfer of the program code**, wherein a number of predefined flags (i.e., static information) are set in order to detect parallelism. As a result of these flags set into the program code, the ILP processor can detect easily when instructions which are to be carried out in parallel occur in the program flow and react appropriately.

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

In a second type of processor, such as in the superscalar architectures of the PowerPC processors and certain "hyper threading" Intel processors, hardware is provided that can detect parallelism **while the program is running**, and then provide an appropriate reaction to it. For example, in Intel "hyper threading technology" processors, a fetch-control unit determines dynamically if process units are idle and if the next instruction in the pipeline can be processed in parallel or not.

However, none of the known prior art devices use both a detection of parallelism during the transfer of the program code (i.e., **statically** set during program code compilation) and the **dynamic** detection of parallelism during the program running time.

Additionally, Applicant's claimed invention operates to execute **at least two different processes**, wherein the instructions of one process **are independent of** the instructions of any second process. As such, the two processes can be executed fully in parallel.

More particularly, Applicant's independent claims recite, among other limitations,

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

a program memory having stored therein at least one compiled program with a multiplicity N of independent processes, the compiled program including information on parallelism and including a multiplicity of bundles with a plurality of instructions of a process, the instructions of a bundle being executable in parallel; [emphasis added by Applicant]

As such, among other things, Applicant's claims require the compiled program to include a multiplicity of bundles, each bundle including a plurality of instructions that can be executed in parallel. As is clear from the claim language, these bundles, and thus the indication of parallelism for the instructions in the bundles, are set at the time the program code is compiled. As such, the determination of the parallelism of the instructions in the bundles is static, and not changed or determined while the program is running. This is additionally supported by the further language of the independent claims, which recite, among other limitations:

a program flow control unit connected to said branching control unit, said program flow control unit controlling a fetching of bundles to be processed in parallel from said program memory, controlling said branching control unit, and controlling an output of instructions to be processed in parallel in dependence on information contained in the instructions and included in a compiling time of the program; [emphasis added by Applicant]

However, Applicant's claimed invention also dynamically determines parallelism while the program is running. In the present invention, bundles are loaded into the instructions buffers 13, 14, such that each bundle (i.e., which contains

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

instructions determined to be executable in parallel at the time of compiling the program) is loaded, in its entirety, into one buffer or the other. Additionally, the two bundles loaded into the buffers **are associated with different processes**. This can be seen from the language of Applicant's independent claims, which recite, among other limitations:

a number N of instruction buffers being connected in parallel downstream of said program memory for storing instructions read out from said program memory, **an instruction bundle being read into one of said instruction buffers and a second instruction bundle associated with a different process being read into another one of said instruction buffers**; [emphasis added by Applicant]

As such, a first bundle of instructions determined to be executable in parallel at the time of compiling are **together** loaded into a first buffer, while a second bundle of instructions determined to be executable in parallel at the time of compiling are **together** loaded into **another buffer**. Note that, the instructions in the first bundle are from a **different process** than the instructions of the second bundle. As such, **the instructions from one bundle in one buffer are executable in parallel with an instruction from another bundle in another buffer**. Applicant's claimed invention **dynamically** determines whether instructions from different bundles (i.e., different processes) are to be executed together and, based on that determination, **selects, in a first case, instructions from different bundles for parallel execution, and, in a**

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

second case, instructions from the same bundle for parallel execution. This can be seen from the language of the independent claims, which recite, among other limitations:

an instruction output selector being connected to and controlled by said program flow control unit causing said instruction output selector to read out instructions from said instruction buffers and output N instructions in parallel, said instruction output selector having a multiplexer logic and **selecting based on a determination by said program flow control unit, in a first case, one instruction from a first instruction buffer and one instruction from a second instruction buffer for execution in parallel and, in a second case, two instructions from one of said first and second instruction buffers for execution in parallel,** [emphasis added by Applicant]

Note that, in the first case, where instructions from **different bundles** are outputted for execution in parallel, this determination must be made **dynamically, i.e., during the running of the program** (i.e., and because of the different processes, need not) as it does not use the static information on parallelism available at the time of compiling the program. As such, Applicants' claimed invention combines the two approaches to parallelism into a single device executing instructions of different processes or the same process in parallel (i.e., by making determinations based on static indications inserted during program compiling **and dynamically** based on information available during the running of the program).

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

That the program flow control unit of Applicant's claim 1 makes a **dynamic** determination during the running of the program is further emphasized by the amended claim 1, which states, among other limitations:

said program flow control unit choosing between said first case or said second case based on one of **priority information, nop information and information of the current process.** [emphasis added by Applicant]

This can also be seen in Applicant's independent claim 13, which recites, among other limitations:

said program flow control unit choosing said first case in order to replace an NOP instruction in one of said first instruction buffer or said second instruction buffer by an instruction from the other buffer. [emphasis added by Applicant]

This can further be seen in Applicant's independent claim 14, which recites, among other limitations:

said program flow control unit choosing between said first case and said second case, based on a priority assigned to each of the different processes.

The above limitations of Applicant's independent claims are supported by the specification of the instant application, for example, page 17, line 14 - page 18, line 8, which states:

The outputting and selection of the instructions from the instruction buffers 13 and 14 and the instruction issue selector 15 is controlled by the flow control unit 10. This will be explained with reference to FIG. 2. The instruction bundles which are read out from the program memory 12 are fed to the instruction buffers

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

13 and 14 via an instruction bus. **The following information from the instruction bundle is fed to the flow control unit 10 via an instruction subbus 25:**

A bit for indicating the parallel execution of instructions or a bit for indicating the length of the following instruction bundle. However, in the case of program code with a fixed length it is not necessary to indicate the length.

The indication of one or more NOPs in the instruction bundles, in which case an NOP can be replaced by another instruction of the other process.

The current process, i.e. the number of the process via a thread bus 28. [emphasis added by Applicant]

The priority of the two processes. [emphasis added by Applicant]

See also, page 18 of the instant application,

Se

As such, the invention of Applicant's independent claims **dynamically use information, during the running of the different processes,** to decide between taking one instruction from each of a first and second buffer, in a first case, or taking two instructions from a single buffer, in a second case, for parallel execution.

Neither a **dynamic determination,** nor the running of **different processes,** both as required by Applicant's claim 1 are disclosed in either of the **HEISHI** or **SUZUKI** references.

First, the **SUZUKI** and **HEISHI** references only disclose an execution of a single process, in contrast to Applicant's

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

claimed "a second instruction bundle associated with a different process being read into another one of said instruction buffers". Further, contrary to Applicant's claimed invention, the cited **SUZUKI**, and **HEISHI** references fail to teach or suggest, utilizing both static and dynamic methods for determining parallelism of instructions. The program code of **SUZUKI** and **HEISHI** is provided with predetermined information on parallelism. For example, **SUZUKI** discloses the use of format specifying bits (FM) encoded in the program for indicating if two instructions can be executed simultaneously and which processor is to execute the instructions. See, col. 2 of **SUZUKI**, lines 14 - 31. However, neither **SUZUKI**, nor **HEISHI**, teaches or suggests dynamically determining which instructions can be executed in parallel during the running of the program, or running as required by Applicant's claims.

The **KAHLE** reference, cited in the Office Action in combination with **SUZUKI** and **HEISHI**, discloses a multiscalar method for parallel processing of multiple tasks. In contrast to the present invention, in **KAHLE**, the tasks are all part of the same program and may exhibit a degree of control and data independence. See, col. 3 of **KAHLE**, lines 8 - 11.

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

As such, the **SUZUKI**, **HEISHI** or **KAHLE** references fail to teach or suggest, among other limitations of Applicant's claims, operating on different processes in parallel. Further, the **SUZUKI**, **HEISHI** or **KAHLE** references fail to teach or suggest, among other limitations of Applicant's claims, using both static and dynamic determination methods to determine whether the instructions from different processes are to be executed in parallel, as claimed by Applicant.

Further, absent impermissible hindsight reconstruction, it would not be obvious to combine the processes of **SUZUKI** and **HEISHI** with that of **KAHLE**. It has long been recognized that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion or motivation to do so found in the references, themselves, or in the knowledge of one of skill in the art. Such a teaching, suggestion or motivation to combine different parallel processing techniques in a single application, in order to execute two independent processes, is absent from the art of record in the instant case.

Further, in multi-scalar devices such as **KAHLE**, each task is assigned to a specific execution unit. Applicant has added

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

claim 15, depending from claim 1, which recites, among other limitations:

The data processing device of claim 1, further comprising **at least two instruction-execution units** for receiving the instructions selected by said instruction output selector, **each of said at least two instruction-execution units being able to execute instructions output by either of said first instruction buffer and said second instruction buffer.**

New claim 15 is supported by the specification of the instant application, for example, page 14 of the instant application, lines 14 - 20, which states:

Two execution units EX1 19 and EX2 20 (instruction execution units) are in turn connected downstream of the register 18. These two units serve to execute the instructions. For this purpose, both units EX1 19 and EX2 20 are each provided with two buses BUS1 21 and BUS2 22 via which a memory 23 in which data is stored is accessed. The memory 23 is preferably a random access memory (RAM).

It can be seen that, since the instructions may both be coming from the same instruction buffer/process, or from different instruction buffers/different processes, in the invention of Applicant's claim 15, **any execution unit can execute any of the instructions.** Contrary to the Applicant's invention of claim 15, in **KAHLE**, a task and its instructions are always executed by the same execution unit. Contrary to **KAHLE**, in Applicant's claim 15, instructions **from a single process** may be loaded and executed by each of the execution units 19, 20 (i.e., by different execution units).

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

In view of the foregoing, Applicant's claims are believed to be patentable over the **SUZUKI**, **HEISHI** and **KAHLE** references, taken alone or in combination. The **ALLEN** reference, cited in combination with **SUZUKI**, **HEISHI** and **KAHLE** against Applicant's dependent claim 12, does not cure the above discussed deficiencies of the **SUZUKI**, **HEISHI** and **KAHLE** references

It is accordingly believed that none of the references, whether taken alone or in any combination, teach or suggest the features of claims 1, 13 and 14. Claims 1, 13 and 14 are, therefore, believed to be patentable over the art. The dependent claims are believed to be patentable as well because they all are ultimately dependent on claim 1.

In view of the foregoing, reconsideration and allowance of claims 1 and 4 - 15 are solicited.

In the event the Examiner should still find any of the claims to be unpatentable, counsel would appreciate receiving a telephone call so that, if possible, patentable language can be worked out.

Applic. No. 09/760,405
Response Dated September 18, 2006
Responsive to Office Action of May 17, 2006

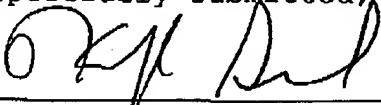
The current Response is being filed simultaneously with a Request for Continued Examination (RCE) and its associated fee.

Additionally, please consider the present as a petition for a one (1) month extension of time, and please provide a one (1) month extension of time, to and including, September 18, 2006, to respond to the present Office Action.

The extension fee for response within a period of one (1) month pursuant to Section 1.136(a) in the amount of \$120.00 in accordance with Section 1.17 is enclosed herewith.

Please provide any additional extensions of time that may be necessary and charge any other fees that might be due with respect to Sections 1.16 and 1.17 to the Deposit Account of Lerner Greenberg Sterner LLP, No. 12-1099.

Respectfully submitted,



Kerry P. Sisselman
Reg. No. 37,237

For Applicant

September 18, 2006

Lerner Greenberg Sterner LLP
Post Office Box 2480
Hollywood, FL 33022-2480
Tel: (954) 925-1100
Fax: (954) 925-1101